CMPT 431: Distributed Systems (Fall 2019)

Assignment 2 - Report

Name	
SFU ID	

Instructions:

- This report is worth 30 points.
- Answer in the space provided.
  Answers spanning beyond 3 lines (11pt font) will lose points.
- Input graphs used are available at the following location.
  - live-journal graph (LJ graph): `/scratch/assignment2/input_graphs/lj`
  - RMAT graph: `/scratch/assignment2/input_graphs/rmat`
- All the experiments are conducted with 4 workers.
- All the times are in seconds.

---

1. [4 points] Run Triangle Counting with `--strategy=1` on the LJ graph and the RMAT graph. Update the thread statistics in the tables below. What is your observation on the difference in time taken by each thread for RMAT and that for LJ? Why does this happen?

Answer:

Triangle Counting on LJ: Total time = 56.49615 seconds.

thread_id	num_vertices	num_edges	triangle_count	time_taken
0	1211892	42920131	357657286	56.49562
1	1211892	15515692	217447326	12.82936
2	1211892	7141449	86161667	3.49256
3	1211895	3416501	46058470	0.95605

Triangle Counting on RMAT: Total time = ___ seconds.

thread_id	num_vertices	num_edges	triangle_count	time_taken
0				
1				
2				
3				

2. [3 points] Run Triangle Counting with `--strategy=2` on LJ graph. Update the thread statistics in the table below. Partitioning time is the time spent on task decomposition as required by `--strategy=2`. What is your observation on the difference in time taken by each thread, and the difference in num_edges for each thread? Are they correlated (yes/no)? Why?

   Answer:

Triangle Counting on LJ: Partitioning time = ___ seconds. Total time = ___ seconds.

thread_id	num_vertices	num_edges	triangle_count	time_taken
0				
1				
2				
3				

3. [1 point] Run Triangle Counting with `--strategy=3` on LJ graph. Update the thread statistics in the table below.

Triangle Counting on LJ: Total time = ___ seconds.

thread_id	num_vertices	num_edges	triangle_count	time_taken
0				
1				
2				
3				

4. [3 points] Run PageRank with `--strategy=1` on LJ graph. Update the thread statistics in the table below. What is your observation on the difference in time taken by each thread, and the difference in num_edges for each thread? Is the work uniformly distributed across threads (yes/no)? Why?

Answer:

____

____

____

PageRank on LJ: Total time = ___ seconds.

thread_id	num_vertices	num_edges	time_taken
0			
1			
2			
3			

5. [3 points] Run PageRank with **`--strategy=1`** on LJ graph. Obtain the cumulative time spent by each thread on **`barrier1`** and **`barrier2`** (refer pagerank pseudocode for program 4 on assignment webpage) and update the table below. What is your observation on the difference in barrier1_time for each thread and the difference in num_edges for each thread? Are they correlated (yes/no)? Why?

   Answer:

   PageRank on LJ: Total time = ___ seconds.

thread_id	num_vertices	num_edges	barrier1_time	barrier2_time	time_taken
0					
1					
2					
3					

6. [3 points] Run PageRank with **`--strategy=2`** on the LJ graph. Update the thread statistics in the table below. Update the time taken for task decomposition as required by **`--strategy=2`**. What is your observation on barrier2_time compared to the barrier2_time in question 5 above? Why are they same/different?

   Answer:

   PageRank on LJ: Total time = ___ seconds. Partitioning time = ___ seconds.

thread_id	num_vertices	num_edges	barrier1_time	barrier2_time	time_taken
0					
1					
2					

3					

7. [3 points] Run PageRank with `--strategy=3` on LJ graph. Update the thread statistics in the table below. What is your observation on barrier times compared to the barrier times in question 6 above? What is your observation on the time taken by each thread compared to time taken by each thread in question 6 above? Why are they same/different?

Answer:

PageRank on LJ: Total time = ___ seconds. Partitioning time = ___ seconds.

thread_id	num_vertices	num_edges	barrier1_time	barrier2_time	time_taken
0					
1					
2					
3					

8. [3 points] Run PageRank with `--strategy=3` on LJ graph. Obtain the total time spent by each thread in `getNextVertexToBeProcessed()` and update the table below. What is your observation on the time taken by `getNextVertexToBeProcessed()`? Why is it high/low?

Answer:

PageRank on LJ: Total time = ___ seconds.

thread_id	num_vertices	num_edges	barrier1_time	barrier2_time	getNextVertex_time	time_taken
0						
1						

2						
3						

9. [3 points] Run PageRank with `--strategy=3` on LJ graph with `--granularity=2000`. Update the thread statistics in the table below. What is your observation on the time taken by `getNextVertexToBeProcessed()`? Why is it high/low?

   Answer:

PageRank on LJ: Granularity = 2000. Total time = ___ seconds. Partitioning time = ___ seconds.

thread_id	num_vertices	num_edges	barrier1_time	barrier2_time	getNextVertex_time	time_taken
0						
1						
2						
3						

10. [4 points] While --strategy=3 with --granularity=2000 performs best across all of our parallel PageRank attempts, it doesn’t give much performance benefits over our serial program (might give worse performance on certain inputs). Why is this the case? How can the parallel solution be improved further to gain more performance benefits over serial PageRank?

    Answer: